**Object-Oriented Programming I (CSE 121)**

**Week # 01**

**(YouTube Lecture: 01, 02 & 03)**

# Introduction to C++ Programming

## Objectives

In this chapter, you will learn:

- What is C++? Why use C++?
- To write simple computer programs in C++
- To write simple input and output statements
- Variable, identifiers and constants
- To use fundamental data types
- To use string data type
- To use operators
- To use manipulators

# Object-Oriented Programming I (CSE 121)
# Week # 01
## Introduction to C++ Programming

## What is C++?

- C++ is a general-purpose programming language. It's used in a wide range of industries including software and game development, VR, robotics, and scientific computing.
- C++ gives programmers a high level of control over system resources and memory.
- C++ was developed by **Bjarne Stroustrup at** Bell Labs in 1979, as an extension to the C language.

## Why Use C++

- C++ is one of the world's most popular programming languages.
- C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.
- C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
- C++ is portable and can be used to develop applications that can be adapted to multiple platforms.

## First Program in C++: Printing a Line of Text

firstProgram.cpp

```cpp
1 #include<iostream>
2 using namespace std;
3
4 int main(){
5     cout << "Welcome to C++";
6     return 0;
7 }
```

C:\Users\Bulbul\Desktop\OOP\firs

```
Welcome to C++
Process returned 0 (0x0)
Press any key to continue.
```

## Example explained

**Line 1:** #include<iostream> is a **header file library** that lets us work with input and output objects, such as cout (used in line 5).

**Line 2:** using namespace std means that we can use names for objects and variables from the standard library.

**Line 3:** A blank line. C++ ignores white space.

**Line 4:** Another thing that always appear in a C++ program, is int main(). This is called a **function**. Any code inside its curly brackets {} will be executed.

**Line 5:** cout (pronounced "see-out") is an **object** used together with the ***insertion operator*** (<<) to output/print text. In our example it will output "Welcome to C++".

**Line 6:** return 0 ends the main function.

**Line 7:** Do not forget to add the closing curly bracket } to actually end the main function.

**Note:** Every C++ statement ends with a semicolon ;.

**Remember:** The compiler ignores white spaces. However, multiple lines makes the code more readable.

## C++ Output (Print Text)

The cout object, together with the << operator, is used to output values/print text.

output.cpp

```
#include<iostream>
using namespace std;

int main() {
    cout << "Welcome to C++";
    cout << "I am learning C++";
    return 0;
}
```

```
C:\Users\Bulbul\Desktop\OOP\output.exe
Welcome to C++I am learning C++
Process returned 0 (0x0)   exec
Press any key to continue.
```

**Note:** You can add as many cout objects as you want. However, note that it does not insert a new line at the end of the output.

## C++ New Lines

To insert a new line, you can use the \n character.

firstProgram.cpp

```
#include<iostream>
using namespace std;

int main() {
    cout << "Welcome to C++\n";
    cout << "I am learning C++\n";
    return 0;
}
```

```
C:\Users\Bulbul\Desktop\OOP\
Welcome to C++
I am learning C++

Process returned 0 (0x0)
```

Another way to insert a new line, is with the endl manipulator.

firstProgram.cpp

```
#include<iostream>
using namespace std;

int main() {
    cout << "Welcome to C++" << endl;
    cout << "I am learning C++" << endl;
    return 0;
}
```

```
C:\Users\Bulbul\Desktop\OOP\
Welcome to C++
I am learning C++

Process returned 0 (0x0)
```

**Note:** Both \n and endl are used to break lines. However, \n is used more often and is the preferred way.

## C++ Comments

- Comments can be used to explain C++ code, and to make it more readable. Comments can be singled-lined or multi-lined.

**Single line comments:**

- Single-line comments start with two forward slashes (`//`).
- Any text between `//` and the end of the line is ignored by the compiler (will not be executed).

**Multi-line comments:**

- Multi-line comments start with `/*` and ends with `*/`.
- Any text between `/*` and `*/` will be ignored by the compiler.

firstProgram.cpp

```
/*
Program: A simple program to display a line of text
Creator: Bulbul Ahamed
Date: June 14, 2020
*/
#include<iostream> //allows program to output data to the screen
using namespace std;

int main(){
    cout << "Welcome to C++"; //Display message
    return 0;
} // end function main
```

***Single or multi-line comments?***

It is up to you which you want to use. Normally, we use `//` for short comments, and `/* */` for longer.

## C++ Variables

Variables are containers for storing data values.

In C++, there are different **types** of variables (defined with different keywords), for example:

- `int`: stores integers (whole numbers), without decimals, such as 123 or -123
- `double`: stores floating point numbers, with decimals, such as 19.99 or -19.99
- `char`: stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- `string`: stores text, such as "Hello World". String values are surrounded by double quotes
- `bool`: stores values with two states: true or false

## Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:

**Syntax:** *type variableName = value;*

Where *type* is one of C++ types (such as `int`), and *variable* is the name of the variable (such as **x** or **num**). The **equal sign (Assignment operator)** is used to assign values to the variable.

Variable.cpp

```cpp
#include<iostream>
using namespace std;

int main(){
    int num = 10; // variable declaration
    cout << num; // displaying value
    return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\Va

```
10
Process returned 0 (0x0)
Press any key to continue.
```

**Note:** The `cout` object is used together with the `<<` operator to display variables.

## C++ Identifiers

All C++ **variables** must be **identified** with **unique names**. These unique names are called **identifiers**.

Identifiers can be short names (like **x** and **y**) or more descriptive names (**age**, **sum**, **totalVolume**).

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits and underscores
- Names must begin with a letter or an underscore (_)
- Names are case sensitive (`myVar` and `myvar` are different variables)
- Names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words (such as `int`) cannot be used as names

## C++ Constants

When you do not want others (or yourself) to override existing variable values, use the `const` keyword (this will declare the variable as "constant", which means **unchangeable and read-only**).

constant.cpp

```cpp
#include<iostream>
using namespace std;
int main(){
    const double PI = 3.1416; //PI will always be 3.1416
    PI = 10; //error: assignment of read-only variable 'PI'
    return 0;
}
```

# C++ User Input

`cin` is a predefined variable that reads data from the keyboard with the extraction operator (`>>`).

In the following example, the user can input a number, which is stored in the variable `num`. Then we print the value of `num`.

```
input.cpp
#include<iostream>
using namespace std;

int main(){
    int num; //Variable declaration
    cout << "Enter a number: ";
    cin >> num; //Get user input from the keyboard
    cout << "Your number is " << num; //Display the input value
    return 0;
}
```

```
C:\Users\Bulbul\Desktop\OOP\inp
Enter a number: 100
Your number is 100
Process returned 0 (0x0)
Press any key to continue.
```

**Add two number:**

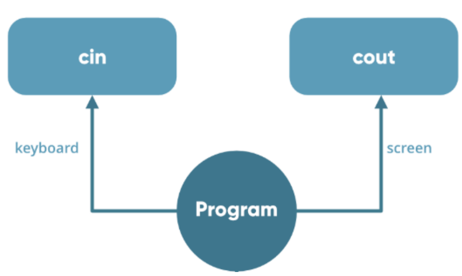
In this program, user is asked to enter two integers. Then, the sum of those two integers is stored in a variable and displayed on the screen.

```
sum.cpp
#include<iostream>
using namespace std;

int main(){
    int x, y, sum;
    cout << "Enter 1st number: ";
    cin >> x;
    cout << "Enter 2nd number: ";
    cin >> y;
        sum = x + y;
    cout << x << " + " << y << " = " << sum;
    return 0;
}
```

```
C:\Users\Bulbul\Desktop\OOP\su
Enter 1st number: 100
Enter 2nd number: 200
100 + 200 = 300
Process returned 0 (0x0)
Press any key to continue.
```

**Note:** The `cin` is a predefined object of **istream** class. The cin is used in conjunction with stream extraction operator (>>) to read the input from a console.




***Good To Know***

`cout` is pronounced "see-out". Used for **output**, and uses the insertion operator (`<<`)

`cin` is pronounced "see-in". Used for **input**, and uses the extraction operator (`>>`)

## C++ Data Types

A variable in C++ must be a specified data type. The data type specifies the size and type of information the variable will store.

To find the size of variable, sizeof operator is used.

sizeOfVariable.cpp

```cpp
#include <iostream>
using namespace std;

int main(){
    cout << sizeof(char) << " byte" << endl;
    cout << sizeof(bool) << " byte" << endl;
    cout << sizeof(int) << " bytes" << endl;
    cout << sizeof(float) << " bytes" << endl;
    cout << sizeof(double) << " bytes" << endl;

    return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\siz

```
1 byte
1 byte
4 bytes
4 bytes
8 bytes

Process returned 0 (0x0)
Press any key to continue.
```

## Basic Data Types

| Data Type | Size | Description |
|---|---|---|
| int | 4 bytes | Stores whole numbers, without decimals |
| float | 4 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 7 decimal digits |
| double | 8 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits |
| boolean | 1 byte | Stores true or false values |
| char | 1 byte | Stores a single character/letter/number, or ASCII values |

dataType.cpp

```cpp
#include <string>
using namespace std;

int main () {
  int myNum = 5;
  float myFloatNum = 5.99;
  double myDoubleNum = 9.98;
  char myLetter = 'D';
  bool myBoolean = true;
  string myString = "Hello";

  cout << "int: " << myNum << "\n";
  cout << "float: " << myFloatNum << "\n";
  cout << "double: " << myDoubleNum << "\n";
  cout << "char: " << myLetter << "\n";
  cout << "bool: " << myBoolean << "\n";
  cout << "string: " << myString << "\n";

  return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\dat

```
int: 5
float: 5.99
double: 9.98
char: D
bool: 1
string: Hello

Process returned 0 (0x0)
Press any key to continue.
```

## C++ Escape Sequences

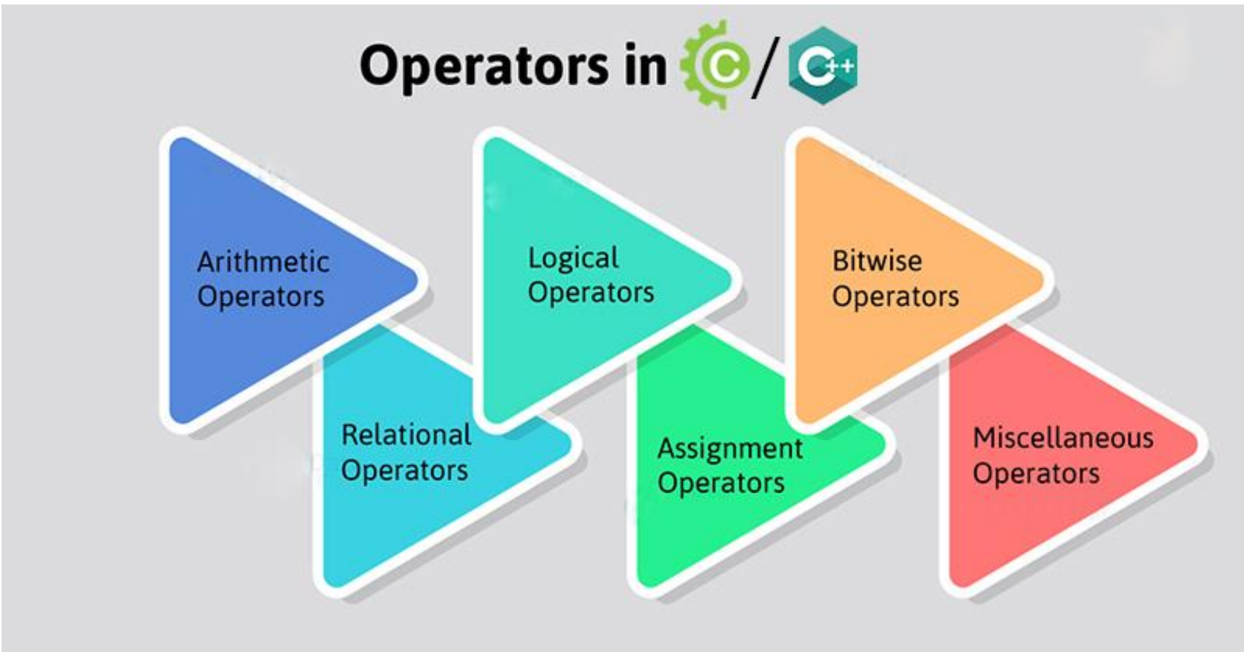
Escape Sequence in C++ Programming Language used to format the output.

**Useful Escape Sequences**

| Escape sequence | Description | Example | Output |
|---|---|---|---|
| \n | New line | printf("Hello \n World"); | Hello<br>World |
| \t | Horizontal tab | printf("Hello \t World"); | Hello World |
| \' | Single quote | printf("Hello \'World\' "); | Hello 'World' |
| \" | Double quote | printf("Hello \"World\" "); | Hello "World" |
| \\ | Backslash | printf("Hello \\World"); | Hello \World |

## C++ Operators

Operators are used to perform operations on variables and values.



## Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

| Operator | Name | Description | Example |
|---|---|---|---|
| + | Addition | Adds together two values | x + y |
| - | Subtraction | Subtracts one value from another | x - y |
| * | Multiplication | Multiplies two values | x * y |
| / | Division | Divides one value by another | x / y |
| % | Modulus | Returns the division remainder | x % y |
| ++ | Increment | Increases the value of a variable by 1 | ++x |
| -- | Decrement | Decreases the value of a variable by 1 | --x |

ArithmeticOperators.cpp

```
#include<iostream>
using namespace std;

int main(){
    int a = 7, b = 2;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "----------" << endl;
    cout << "a + b = " << a + b << endl;
    cout << "a - b = " << a - b << endl;
    cout << "a * b = " << a * b << endl;
    cout << "a / b = " << a / b << endl;
    cout << "a % b = " << a % b << endl;
    return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\Ari

```
a = 7
b = 2
----------
a + b = 9
a - b = 5
a * b = 14
a / b = 3
a % b = 1

Process returned 0 (0x0)
Press any key to continue.
```

## Increment / Decrement operators example

IncrementDecrement.cpp

```
#include<iostream>
using namespace std;

int main(){
    int a = 10;
    cout << "a = " << a << endl;
    cout << "----------" << endl;
    cout << "a++ = " << a++ << endl;
    cout << "++a = " << ++a << endl;
    cout << "--a = " << --a << endl;
    cout << "a-- = " << a-- << endl;
    return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\In

```
a = 10
----------
a++ = 10
++a = 12
--a = 11
a-- = 11

Process returned 0 (0x0)
Press any key to continue.
```

## Relational operators / Comparison Operators

Relational operators are used to compare two values.

| Operator | Name | Example |
|---|---|---|
| == | Equal to | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

**Note:** The return value of a comparison is either true (`1`) or false (`0`).

```cpp
#include<iostream>
using namespace std;

int main(){
    int a = 3, b = 5;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "-----------" << endl;
    cout << "a == b is " << (a == b) << endl;
    cout << "a != b is " << (a != b) << endl;
    cout << "a > b is " << (a > b) << endl;
    cout << "a < b is " << (a < b) << endl;
    cout << "a >= b is " << (a >= b) << endl;
    cout << "a <= b is " << (a <= b) << endl;
    return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\Rel

```
a = 3
b = 5
-----------
a == b is 0
a != b is 1
a > b is 0
a < b is 1
a >= b is 0
a <= b is 1

Process returned 0 (0x0)
Press any key to continue.
```

## Logical Operators

Logical operators are used if we want to compare more than one condition.

<table>
<tr><th>Operator</th><th>Name</th><th>Description</th><th>Example</th></tr>
<tr><td>&&</td><td>Logical and</td><td>Returns true if both statements are true</td><td>x < 5 && x < 10</td></tr>
<tr><td>||</td><td>Logical or</td><td>Returns true if one of the statements is true</td><td>x < 5 || x < 4</td></tr>
<tr><td>!</td><td>Logical not</td><td>Reverse the result, returns false if the result is true</td><td>!(x < 5 && x < 10)</td></tr>
</table>

LogicalOperators.cpp

```cpp
#include <iostream>
using namespace std;

int main() {

    cout << "(3 != 5) && (3 < 5) is " << ((3 != 5) && (3 < 5)) << endl;
    cout << "(3 == 5) && (3 < 5) is " << ((3 == 5) && (3 < 5)) << endl;
    cout << "(3 != 5) || (3 > 5) is " << ((3 != 5) || (3 > 5)) << endl;
    cout << "(3 == 5) || (3 > 5) is " << ((3 == 5) || (3 > 5)) << endl;
    cout << "!(5 == 2) is " << !(5 == 2) << endl;
    cout << "!(5 == 5) is " << !(5 == 5) << endl;

    return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\L

```
(3 != 5) && (3 < 5) is 1
(3 == 5) && (3 < 5) is 0
(3 != 5) || (3 > 5) is 1
(3 == 5) || (3 > 5) is 0
!(5 == 2) is 1
!(5 == 5) is 0

Process returned 0 (0x0)
Press any key to continue
```

## Assignment Operators

Assignment operators are used to assign values to variables.

| Operator | Example | Equivalent to |
|---|---|---|
| `=` | `a = b;` | `a = b;` |
| `+=` | `a += b;` | `a = a + b;` |
| `-=` | `a -= b;` | `a = a - b;` |
| `*=` | `a *= b;` | `a = a * b;` |
| `/=` | `a /= b;` | `a = a / b;` |
| `%=` | `a %= b;` | `a = a % b;` |

AssignmentOperators.cpp

```cpp
#include <iostream>
using namespace std;

int main() {

    int a = 7;
    cout << "a = " << a << endl;
    cout << "------------" << endl;
    cout << "a += 5 is " << (a += 5) << endl;
    cout << "a -= 2 is " << (a -= 2) << endl;
    cout << "a *= 3 is " << (a *= 3) << endl;
    cout << "a /= 4 is " << (a /= 4) << endl;
    cout << "a %= 2 is " << (a %= 2) << endl;

    return 0;
}
```

C:\Users\Bulbul\Desktop\OOP\As

```
a = 7
------------
a += 5 is 12
a -= 2 is 10
a *= 3 is 30
a /= 4 is 7
a %= 2 is 1

Process returned 0 (0x0)
Press any key to continue.
```

## Bitwise Operators

Bitwise operators perform operations on integer data at the individual bit-level.

| Operator | Description |
|---|---|
| & | Bitwise AND Operator |
| \| | Bitwise OR Operator |
| ^ | Bitwise XOR Operator |
| ~ | Bitwise Complement Operator |
| << | Bitwise Shift Left Operator |
| >> | Bitwise Shift Right Operator |

bitwiseOperators.cpp

```cpp
#include<iostream>
using namespace std;

int main(){
    int a = 4, b = 5;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "-----------\n";
    cout << "a & b is " << (a & b) << endl;
    cout << "a | b is " << (a | b) << endl;
    cout << "a ^ b is " << (a ^ b) << endl;
    cout << "a << 1 is " << (a << 1) << endl;
    cout << "a << 2 is " << (a << 2) << endl;
    cout << "a >> 1 is " << (a >> 1) << endl;
    cout << "a >> 2 is " << (a >> 2) << endl;
    return 0;
}
```

C:\Users\Bulbul\Deskto

```
a = 4
b = 5
-----------
a & b is 4
a | b is 5
a ^ b is 1
a << 1 is 8
a << 2 is 16
a >> 1 is 2
a >> 2 is 1

Process returned 0
```

Here, a = 4 = 100(Binary Equivalent) and b = 5 = 101(Binary Equivalent)

| | | |
|---|---|---|
| a = 4 = 100<br>b = 5 = 101 | a = 4 = 100<br>b = 5 = 101 | a = 4 = 100<br>b = 5 = 101 |
| a & b = 100 (4) | a \| b = 101 (5) | a ^ b = 001 (1) |

Bitwise Left-shift & Bitwise Right-shift

**a << 1**: a = 4 = 100 = 100**0** = 8 **a << 2**: a = 4 = 100 = 100**00** = 16

**a >> 1**: a = 4 = 100 = 10~~0~~ = 10 = 2 **a >> 2**: a = 4 = 100 = 1~~00~~ = 1 = 1

## Miscellaneous Operators / Misc Operators

Besides the operators discussed above, there are a few other important operators including **sizeof** and **?:** supported by the C Language.

| Operator | Operation | Description |
|---|---|---|
| sizeof | sizeof(a) | It returns the memory occupied by the particular data type of the operand |
| & | &a | It refers to the address (memory location) in which the operand is stored. |
| * | *a | It is a pointer |
| ?: | a? b: statement | It is an alternative for if-else condition |

**Program:** Find the largest between two numbers using ternary operator (?:)

ternaryOperator.cpp

```cpp
#include<iostream>
using namespace std;

int main(){
    int n1 = 50, n2 = 100, max;
    max = (n1 > n2) ? n1 : n2;
    cout << "Largest number = " << max << endl;
    return 0;
}
```

```
Largest number = 100

Process returned 0 (0x0)
Press any key to continue
```

## C++ Manipulators

Manipulators are functions and operators used in C++ for **formatting output**.

**Some important manipulator functions in <iomanip> are:**

1. **setw(val)**: It is used to set the minimum field width in output operations.
2. **setprecision(val)**: It sets the total number of digits to be displayed, when floating point numbers are printed.
3. **setfill(c)**: It is used to fill the character 'c' on output stream.
4. **setbase(base):** It is used to convert the base field of one numeric value into another base. Possible values are:
   - **dec**: decimal base (base = 10)
   - **hex**: hexadecimal base (base = 16)
   - **oct**: octal base (base = 8)

**Some important manipulator operators in <ios> are:**

1. **endl**: It is used to insert a new line character and flush the stream.
2. **fixed**: It is used to set the number of digits printed to the right of the decimal point.
3. **left**: It adjusts output to the left.
4. **right**: It adjusts output to the right.

**Program:** Using endl, left, right, and setw(value) manipulators

```
#include<iostream>
#include<iomanip>
using namespace std;

int main(){
   int a = 1, ab = 10, abc = 100, abcd = 1000;

   cout << a << endl;
   cout << ab << endl;
   cout << abc << endl;
   cout << abcd << endl;
   cout << "--------------" << endl;

   cout << setw(4) << a << endl;
   cout << setw(4) << ab << endl;
   cout << setw(4) << abc << endl;
   cout << setw(4) << abcd << endl;
   cout << "--------------" << endl;

   cout << "a = " << a << endl;
   cout << "ab = " << ab << endl;
   cout << "abc = " << abc << endl;
   cout << "abcd = " << abcd << endl;
   cout << "--------------" << endl;

   cout << setw(7) << "a = " << a << endl;
   cout << setw(7) << "ab = " << ab << endl;
   cout << setw(7) << "abc = " << abc << endl;
   cout << setw(7) << "abcd = " << abcd << endl;
   cout << "--------------" << endl;

   cout << setw(7) << "a = " << setw(5) << a << endl;
   cout << setw(7) << "ab = " << setw(5) << ab << endl;
   cout << setw(7) << "abc = " << setw(5) << abc << endl;
   cout << setw(7) << "abcd = " << setw(5) << abcd << endl;
   cout << "--------------" << endl;

   cout << left << setw(7) << "a = " << right << setw(5) << a << endl;
   cout << left << setw(7) << "ab = " << right << setw(5) << ab << endl;
   cout << left << setw(7) << "abc = " << right << setw(5) << abc << endl;
   cout << left << setw(7) << "abcd = " << right << setw(5) << abcd << endl;
   cout << "--------------" << endl;
   return 0;
}
```

## Output:

```
C:\Users\Bulbul\Desktop\Lect
1
10
100
1000
--------------
   1
  10
 100
1000
--------------
a = 1
ab = 10
abc = 100
abcd = 1000
--------------
   a = 1
  ab = 10
 abc = 100
abcd = 1000
--------------
   a =      1
  ab =     10
 abc =    100
abcd =   1000
--------------
a =         1
ab =       10
abc =     100
abcd =   1000
--------------
```

**Program:** Using fixed, and setprecision(value) manipulators

```cpp
#include<iostream>
#include<iomanip>
using namespace std;

int main(){
    double a = 3.5619;
    cout << fixed << setprecision(1)<< a << endl;
    cout << fixed << setprecision(2)<< a << endl;
    cout << fixed << setprecision(3)<< a << endl;
    cout << fixed << setprecision(4)<< a << endl;
    cout << fixed << setprecision(5)<< a << endl;
    cout << fixed << setprecision(6)<< a << endl;
    return 0;
}
```

```
C:\Users\Bulbul\Deskto
3.6
3.56
3.562
3.5619
3.56190
3.561900

Process returned 0
```

**Program:** Number conversion using setbase(value) manipulator

three.cpp

```cpp
#include<iostream>
#include<iomanip>
using namespace std;

int main(){
    int a = 15;
    cout << "D => O : " << setbase(8) << a << endl;
    cout << "D => H : " << setbase(16) << a << endl;
    cout << "O => D : " << setbase(10) << 017 << endl;
    cout << "H => D : " << setbase(10) << 0xf << endl;
    return 0;
}
```

C:\Users\Bulb

```
D => O : 17
D => H : f
O => D : 15
H => D : 15
```

**Program:** Uses of setfill(‘c’) manipulator

setfill.cpp

```cpp
#include<iostream>
#include<iomanip>
using namespace std;

int main(){
    int a = 12345, b = 123;
    cout << setfill('*');
    cout << setw(7) << a << endl;
    cout << setw(7) << b << endl;
    return 0;
}
```

C:\Users\Bu

```
**12345
****123
```

.....................